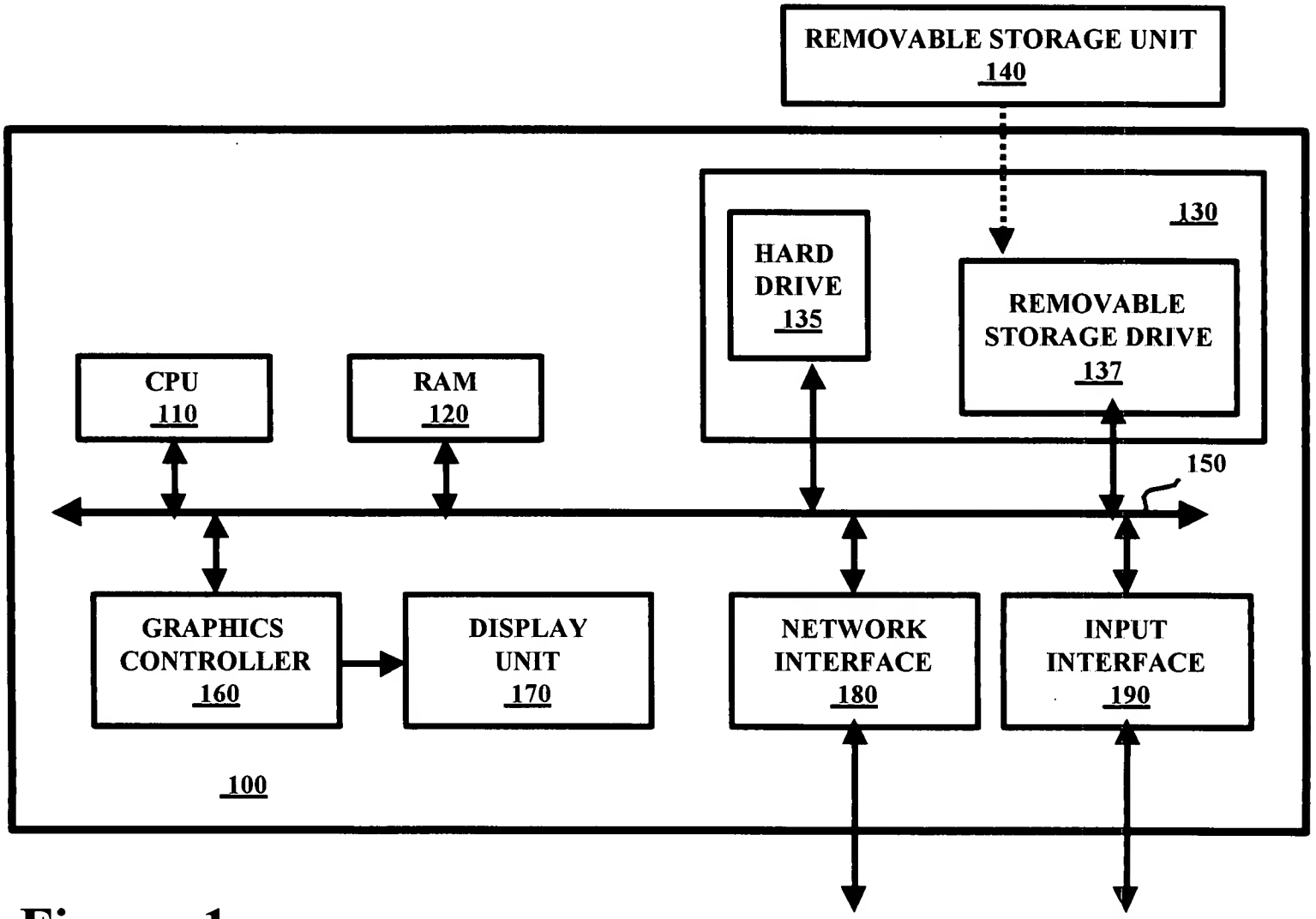


Figure 1

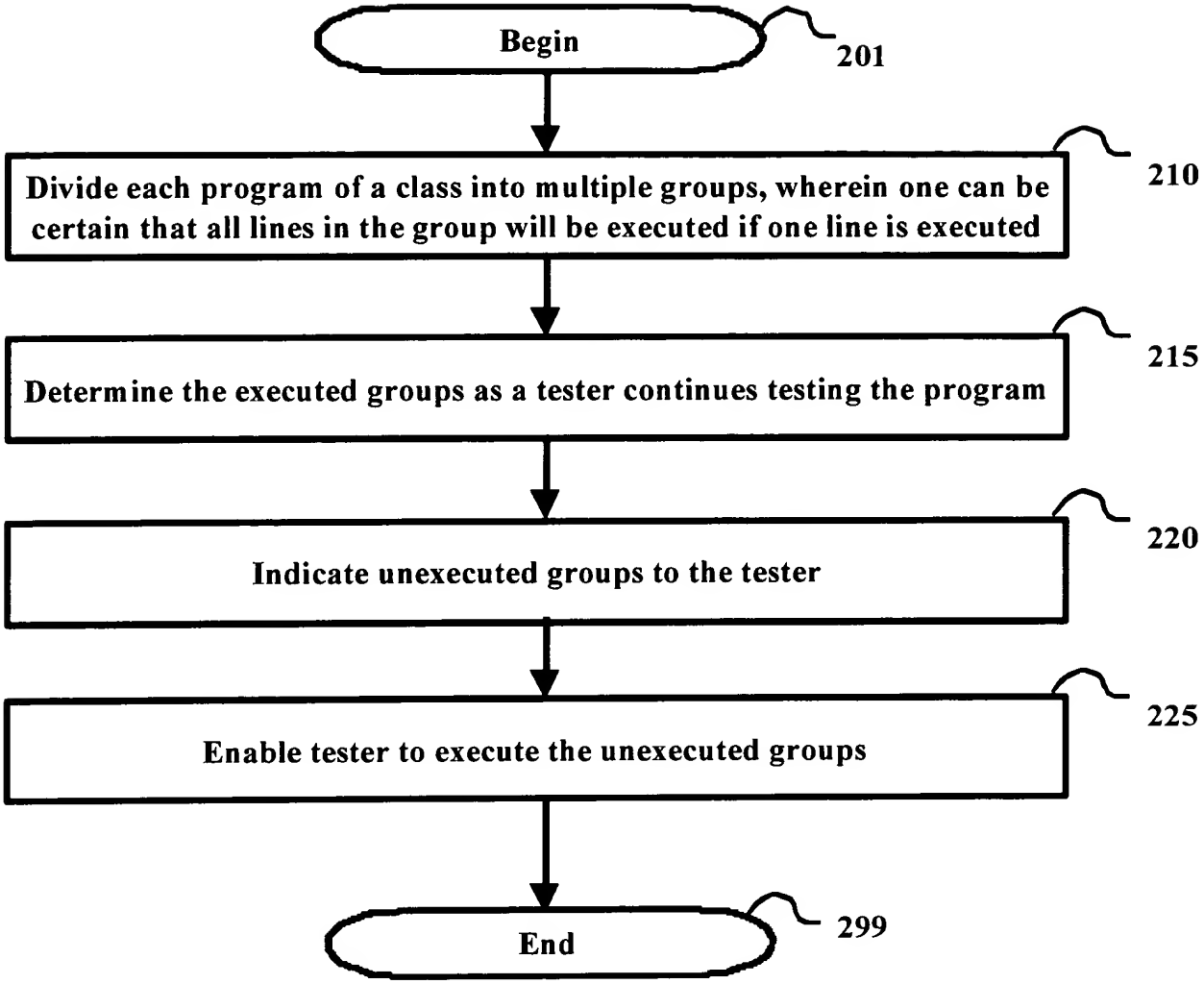

Begin
201
Divide each program of a class into multiple groups, wherein one can be certain that all lines in the group will be executed if one line is executed
210
Determine the executed groups as a tester continues testing the program
215
Indicate unexecuted groups to the tester
220
Enable tester to execute the unexecuted groups
225
End
299

Figure 2

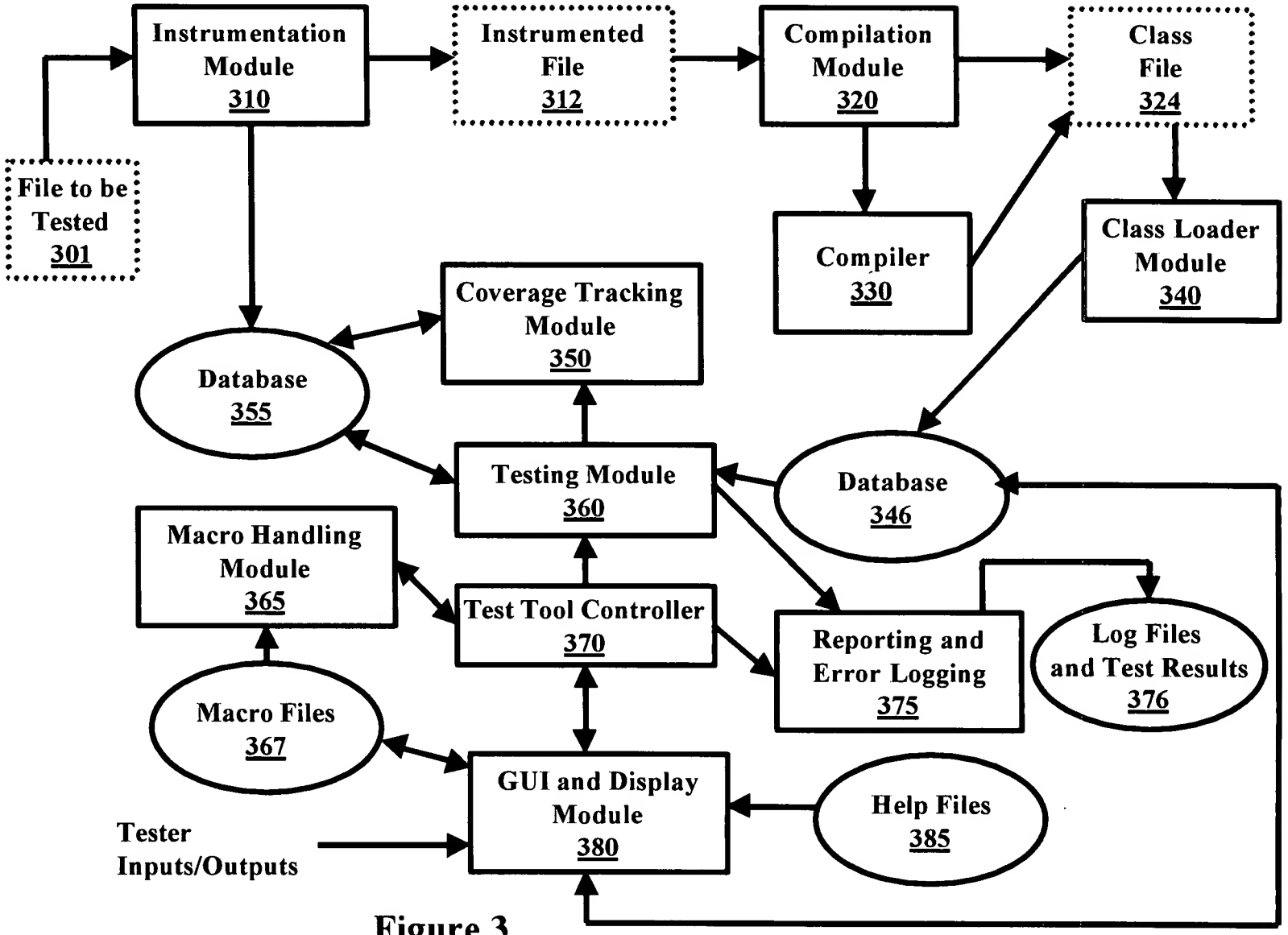

Instrumentation Module 310
Instrumented File 312
Compilation Module 320
Class File 324
File to be Tested 301
Compiler 330
Class Loader Module 340
Coverage Tracking Module 350
Database 355
Testing Module 360
Database 346
Macro Handling Module 365
Test Tool Controller 370
Reporting and Error Logging 375
Log Files and Test Results 376
Macro Files 367
GUI and Display Module 380
Help Files 385
Tester Inputs/Outputs

Figure 3

```
1   public void addString(String inStr) {
2           /* The location where the string needs to be set */
3           int setPos = -1;
4                   /* Check the size to see if the next set needs to be allocated */
5           if( list.size() > 0 && (
6   (String)(list.elementAt(list.size()-1))).equals(emptyString) ) {
7                   /* We still have empty space, get the first empty slot */
8                   int totalRec = list.size();
9                   for(int index=totalRec-1;index >= 0;index--) {
10                      if(!((String)list.elementAt(index)).equals(emptyString)) {
11                              break;
12                      }
13                      else {
14                              setPos = index;
15                      }
16                  }
17          }
18          else {
19                  /* We have run out of space, allocate the nest set */
20                  int size = list.size();
21                  for(int index=0;index<numSetVal;index++) {
22                          list.addElement(emptyString);
23                  }
24                  setPos = size;
25          }

26          /* Add the element */
27          list.setElementAt(inStr,setPos);
28  }
```

Figure 4A

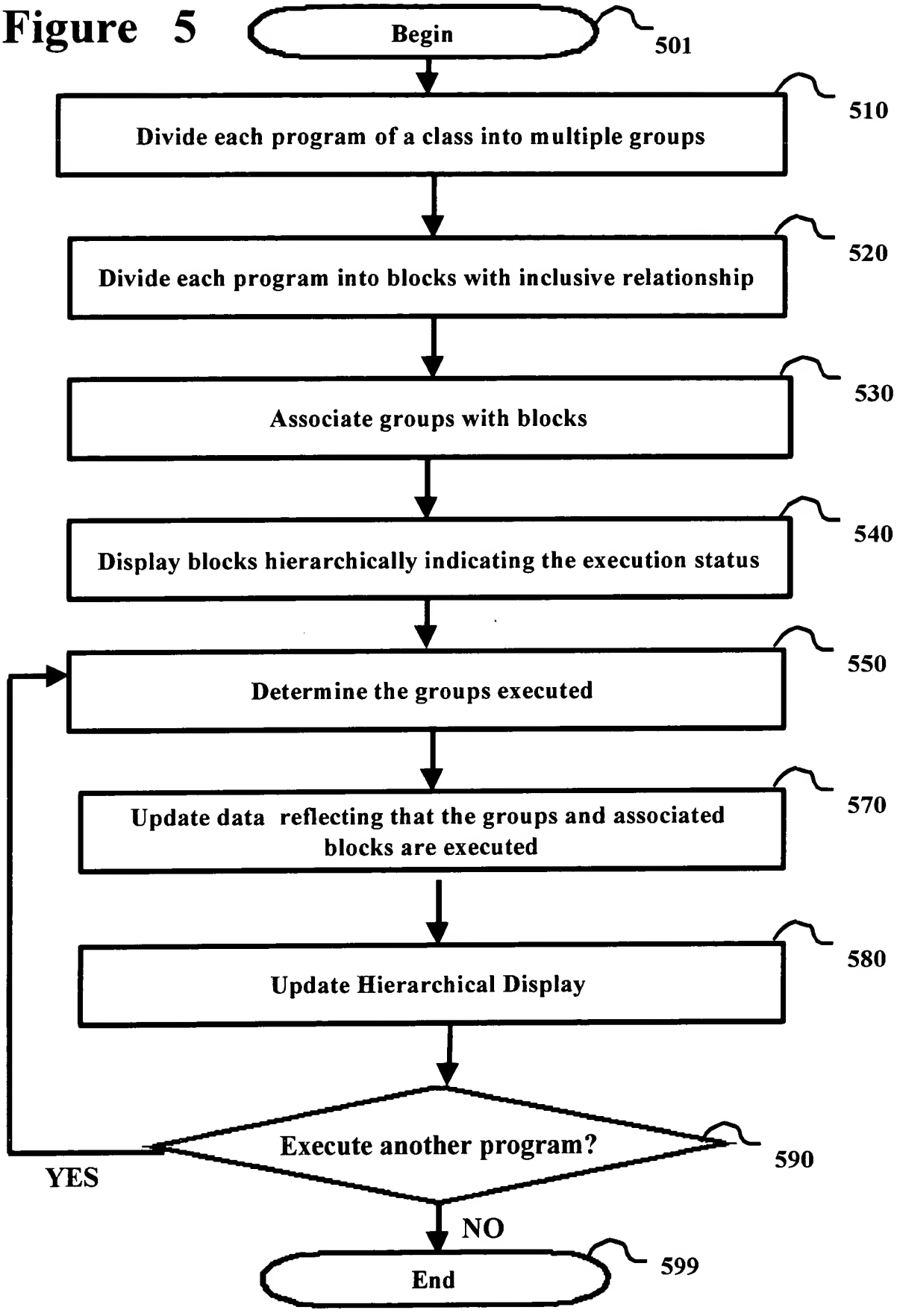
```
1   public void addString(String inStr) {
2           /* INTRUMENTATION Group B11 - BEGIN */
3           Wiz_Tracer.trackExecution("WizStringList",11);
4           /* INTRUMENTATION GROUP B11 - END */
5           int setPos = -1;
6           if( list.size() > 0 && (
7   (String)(list.elementAt(list.size()-1))).equals(emptyString) )
8   {/*INTRUMENTATION GROUP B12 - BEGIN */
9                   Wiz_Tracer.trackExecution("WizStringList",12);
10                  /* INTRUMENTATION GROUP B12 - END */
11                  int totalRec = list.size();
12                  for(int index=totalRec-1;index >= 0;index--) {
13                          /* INTRUMENTATION GROUP B13 - BEGIN */
14                          Wiz_Tracer.trackExecution("WizStringList",13);
15                          /* INTRUMENTATION GROUP B13 - END */
16                          if(!((String)list.elementAt(index)).equals(emptyString)) {
17                                  /* INTRUMENTATION GROUP B14 - BEGIN */
18                                  Wiz_Tracer.trackExecution("WizStringList",14);
19                                  /* INTRUMENTATION GROUP B14 - END */
20                                  break;
21                          }       /* <if GROUP(B14) ends at line : 134> */
```

Figure 4B

```
22                          else {    /* <else GROUP(B15) begins at line : 135> */
23                                    /* INTRUMENTATION GROUP B15 - BEGIN */
24                                    Wiz_Tracer.trackExecution("WizStringList",15);
25                                    /* INTRUMENTATION GROUP B15 - END */
26                                    setPos = index;
27      }         /* <else block(B15) ends at line : 138> */
28                          /* <break(B16) begins at line : 139> */
29                          /* INTRUMENTATION GROUP B16 - BEGIN */
30                          Wiz_Tracer.trackExecution("WizStringList",16);
31                          /* INTRUMENTATION GROUP B16 - END */
32                  }       /* <for block(B13) ends at line : 139> */
33              }       /* <if block(B12) ends at line : 141> */
34              else {  /* <else block(B17) begins at line : 142> */
35                      /* INTRUMENTATION GROUP B17 - BEGIN */
36                      Wiz_Tracer.trackExecution("WizStringList",17);
37                      /* INTRUMENTATION GROUP B17 - END */
38                      int size = list.size();
39                      for(int index=0;index<numSetVal;index++) {      /* <for
40                                      block(B18) begins at line : 146> */
41                              /* INTRUMENTATION GROUP B18 - BEGIN */
42                              Wiz_Tracer.trackExecution("WizStringList",18);
43                              /* INTRUMENTATION GROUP B18 - END */
44                              list.addElement(emptyString);
45                      }       /* <for block(B18) ends at line : 149> */
46                      setPos = size;
47              }       /* <else block(B17) ends at line : 152> */
48              list.setElementAt(inStr,setPos);
49      }       /* <method block - addString(B11) ends at line : 156> */
```

Figure 4C

Figure 5


Begin
501
Divide each program of a class into multiple groups
510
Divide each program into blocks with inclusive relationship
520
Associate groups with blocks
530
Display blocks hierarchically indicating the execution status
540
Determine the groups executed
550
Update data reflecting that the groups and associated blocks are executed
570
Update Hierarchical Display
580
Execute another program?
590
YES
NO
End
599

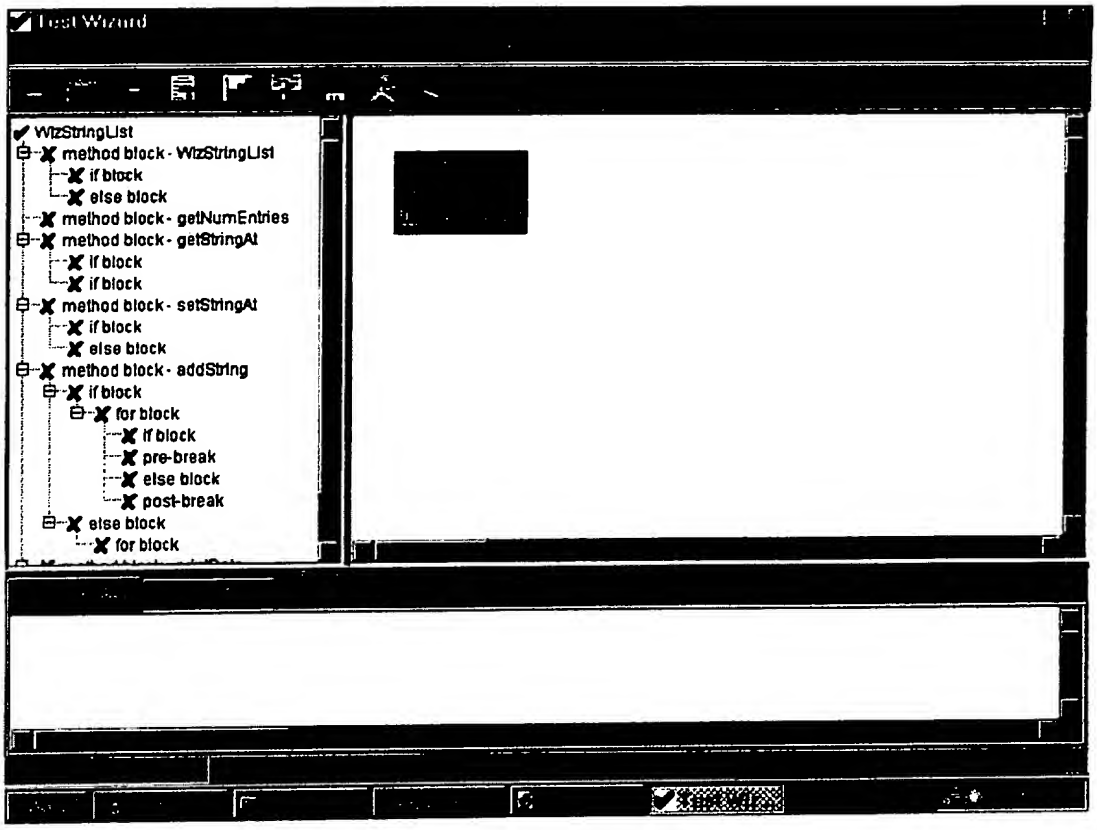

Test Wizard
WizStringList
method block - WizStringList
if block
else block
method block - getNumEntries
method block - getStringAt
if block
if block
method block - setStringAt
if block
else block
method block - addString
if block
for block
if block
pre-break
else block
post-break
else block
for block

Figure 6

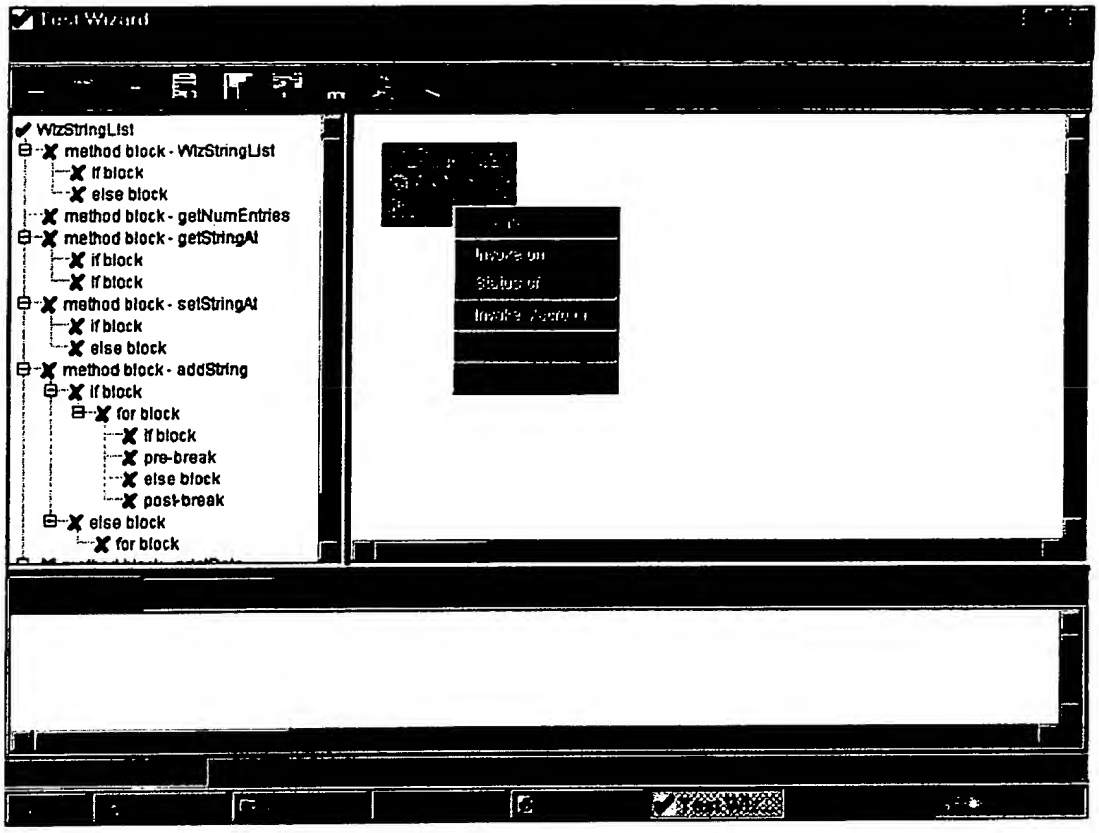

Test Wizard
WizStringList
method block - WizStringList
if block
else block
method block - getNumEntries
method block - getStringAt
if block
if block
method block - setStringAt
if block
else block
method block - addString
if block
for block
if block
pre-break
else block
post-break
else block
for block

Figure 7A

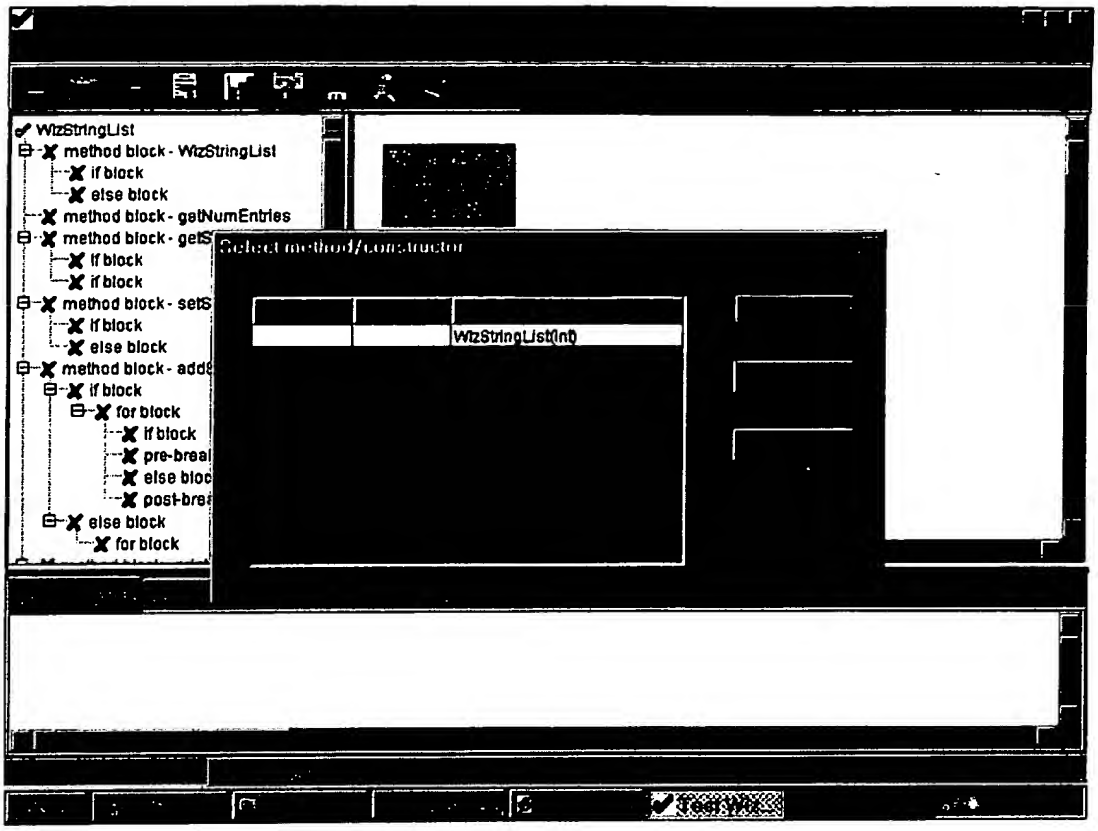

WizStringList
method block - WizStringList
if block
else block
method block - getNumEntries
if block
if block
else block
if block
else block
for block
else block
for block
WizStringList(int)

Figure 7B

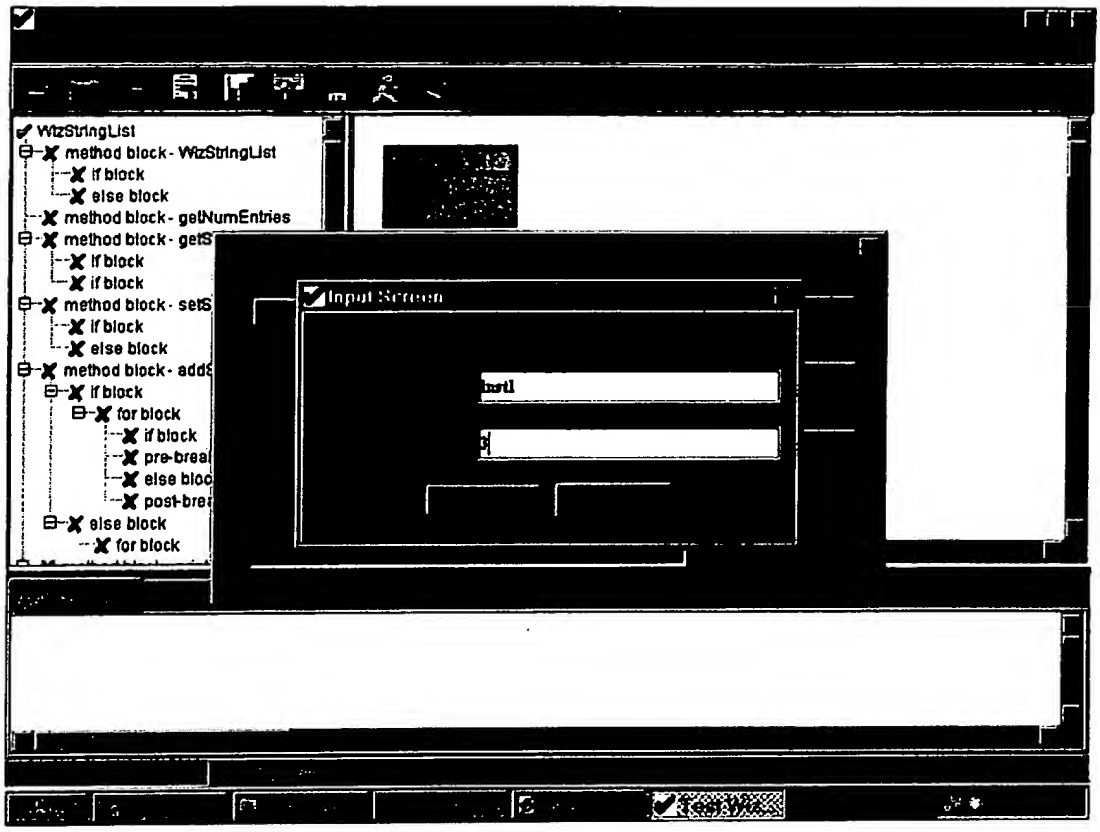

WizStringList
method block - WizStringList
if block
else block
method block - getNumEntries
if block
if block
else block
if block
for block
else block
for block
Input Screen

Figure 7C

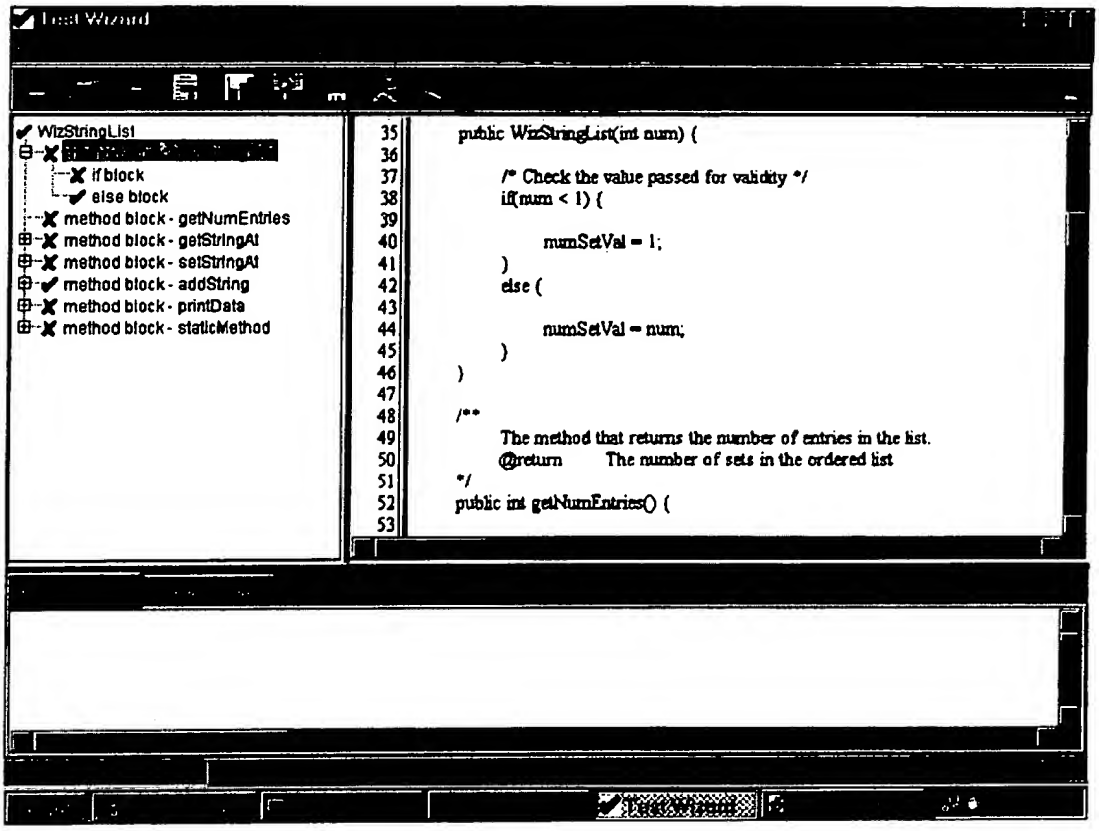

Test Wizard
WizStringList
if block
else block
method block - getNumEntries
method block - getStringAt
method block - setStringAt
method block - addString
method block - printData
method block - staticMethod
35 public WizStringList(int num) {
36
37 /* Check the value passed for validity */
38 if(num < 1) {
39
40 numSetVal = 1;
41 }
42 else {
43
44 numSetVal = num;
45 }
46 }
47
48 /**
49 The method that returns the number of entries in the list.
50 @return The number of sets in the ordered list
51 */
52 public int getNumEntries() {
53

Figure 8

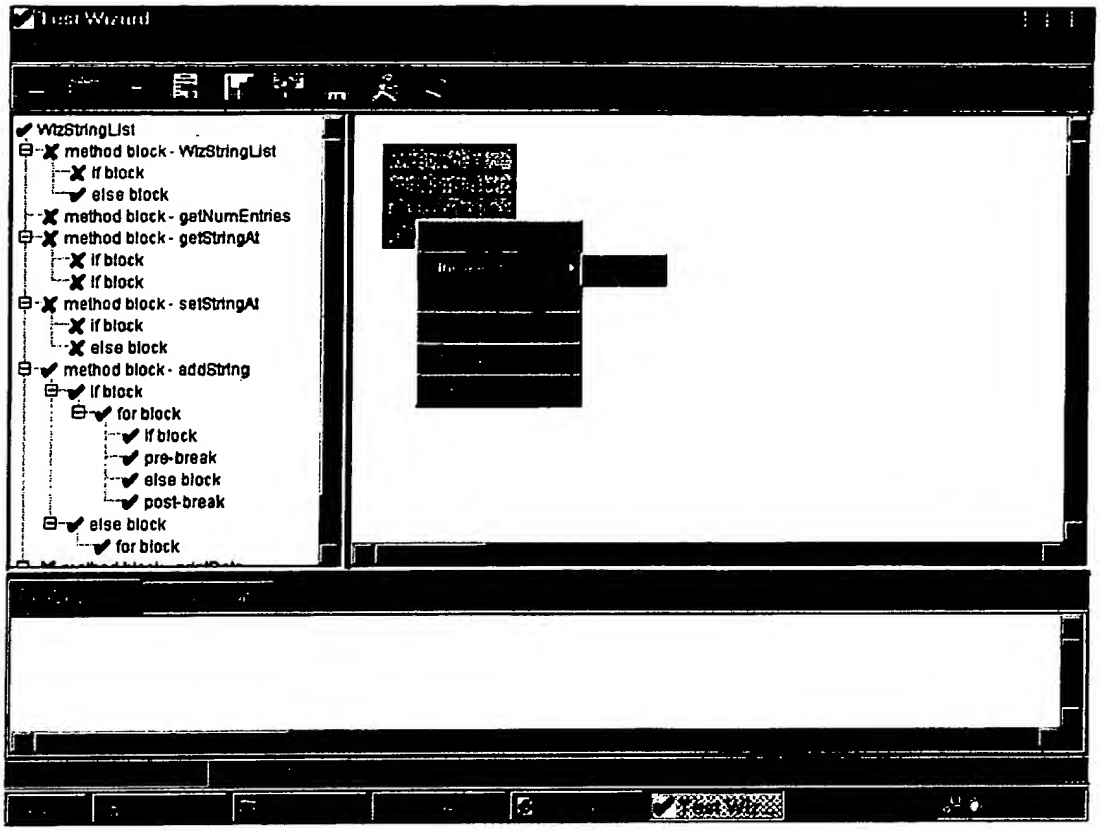

WizStringList
method block - WizStringList
if block
else block
method block - getNumEntries
method block - getStringAt
if block
if block
method block - setStringAt
if block
else block
method block - addString
if block
for block
if block
pre-break
else block
post-break
else block
for block

Figure 9A

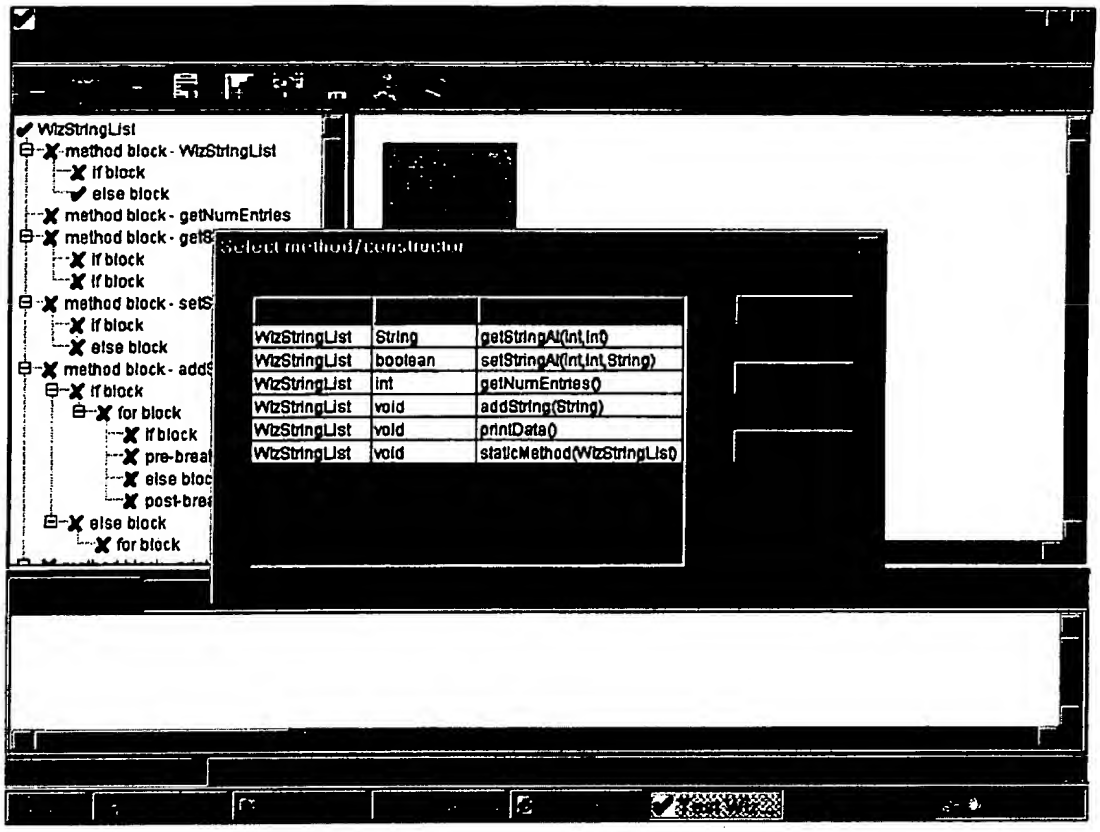

WizStringList
method block - WizStringList
if block
else block
method block - getNumEntries
if block
if block
if block
else block
if block
for block
if block
else block
for block
Select method/constructor
WizStringList String getStringAt(int,int)
WizStringList boolean setStringAt(int,int,String)
WizStringList int getNumEntries()
WizStringList void addString(String)
WizStringList void printData()
WizStringList void staticMethod(WizStringList)

Figure 9B

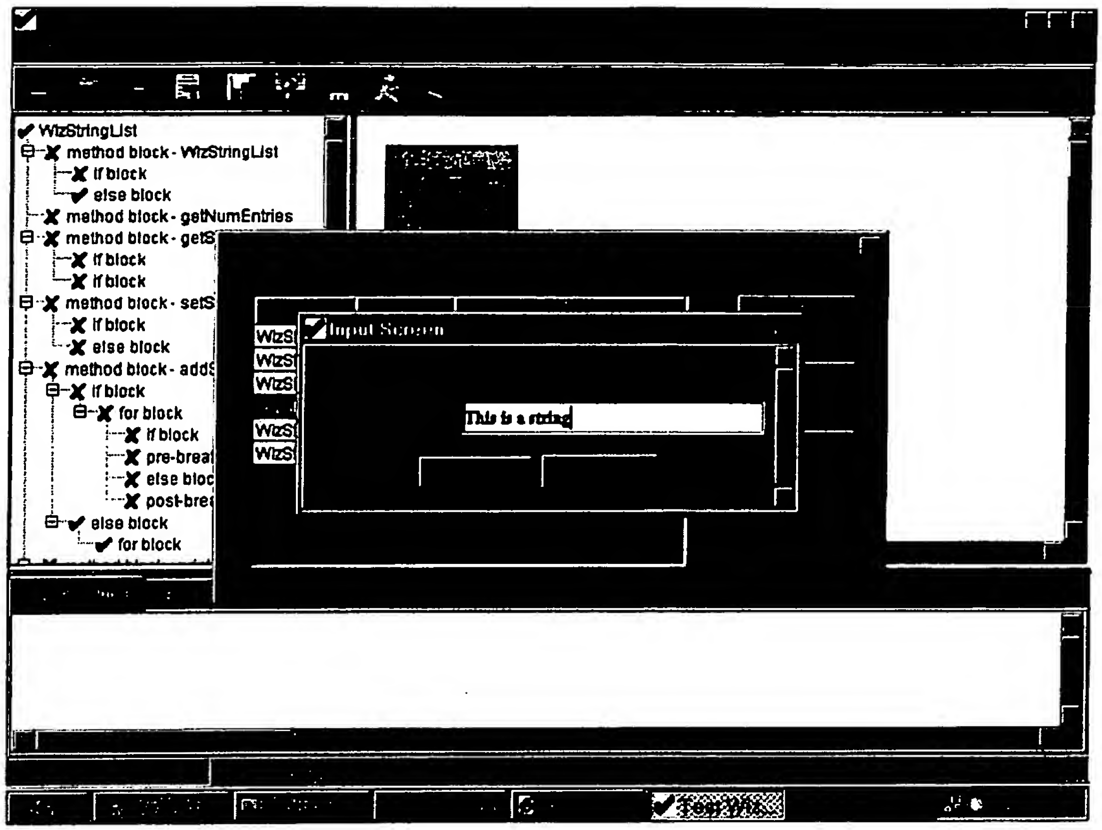

WizStringList
method block - WizStringList
if block
else block
method block - getNumEntries
if block
if block
if block
else block
for block
if block
else block
for block
Input Screen
This is a string

Figure 9C

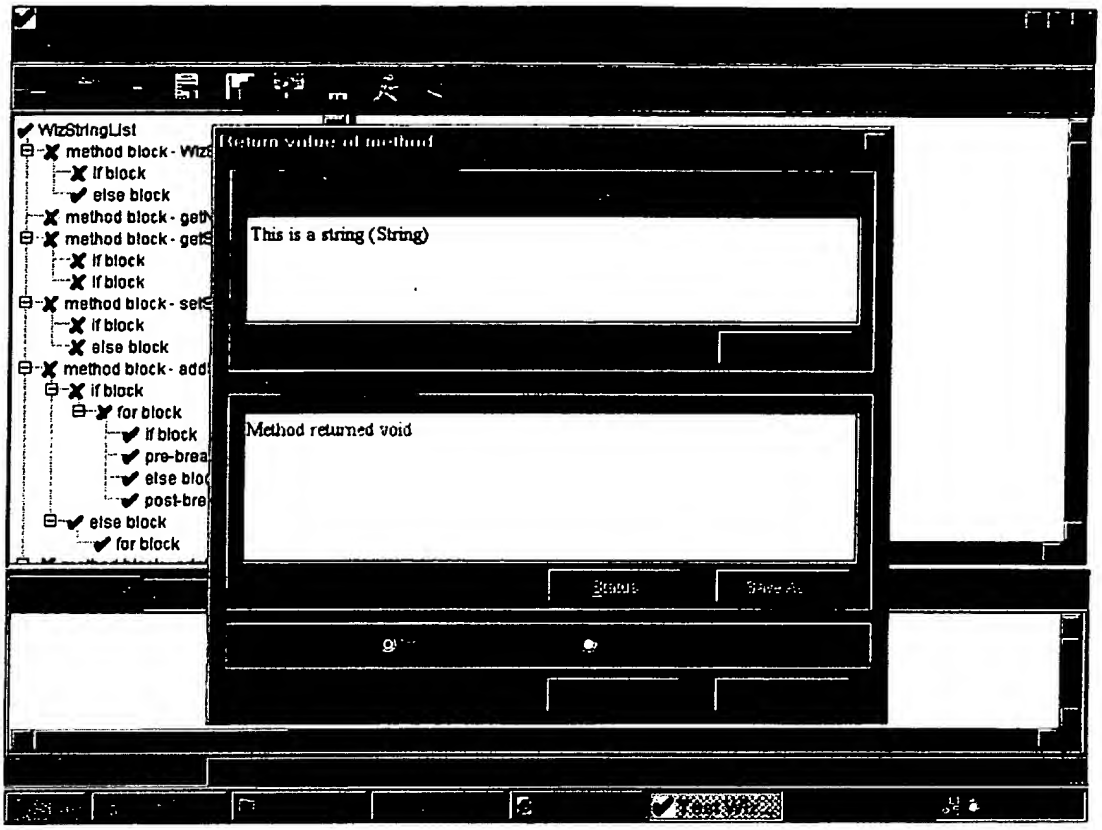

WizStringList
method block - Wiz
if block
else block
method block - geth
method block - get
if block
if block
method block - set
if block
else block
method block - add
if block
for block
if block
pre-brea
else bloc
post-bre
else block
for block
Return value of method
This is a string (String)
Method returned void

Figure 9D

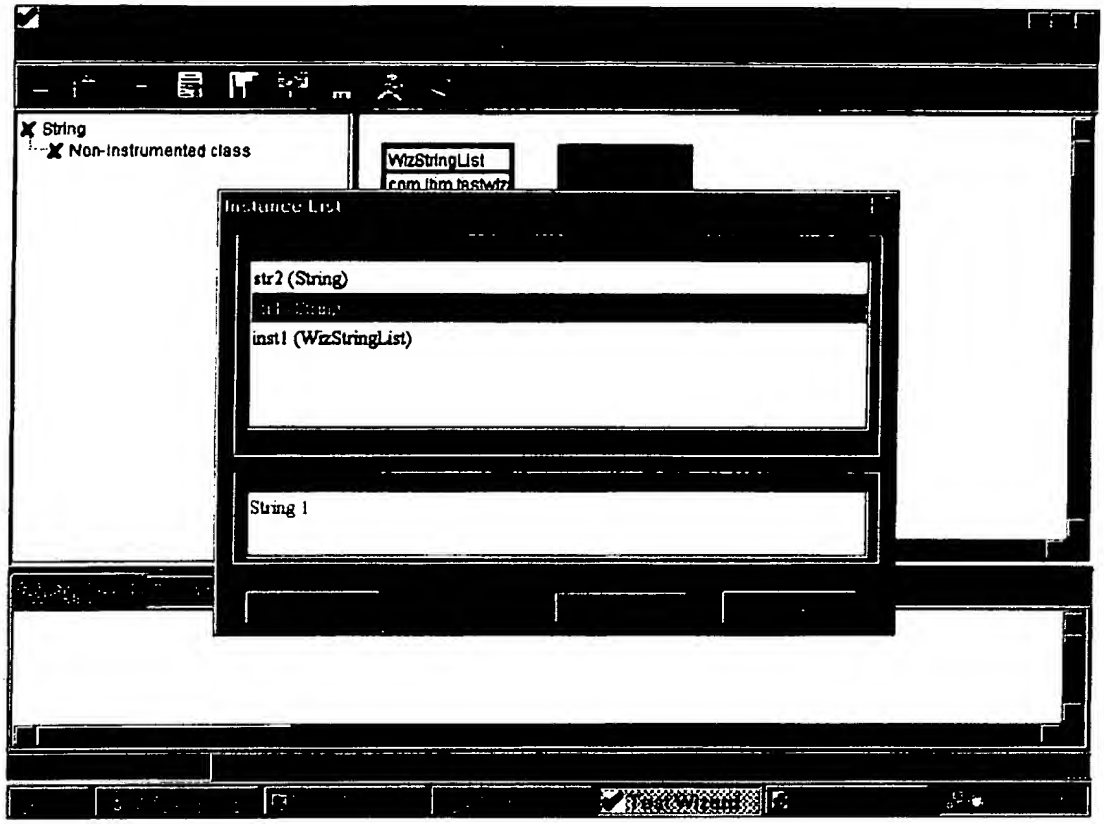

String
Non-instrumented class
WizStringList
Instance List
str2 (String)
inst1 (WizStringList)
String 1

Figure 10

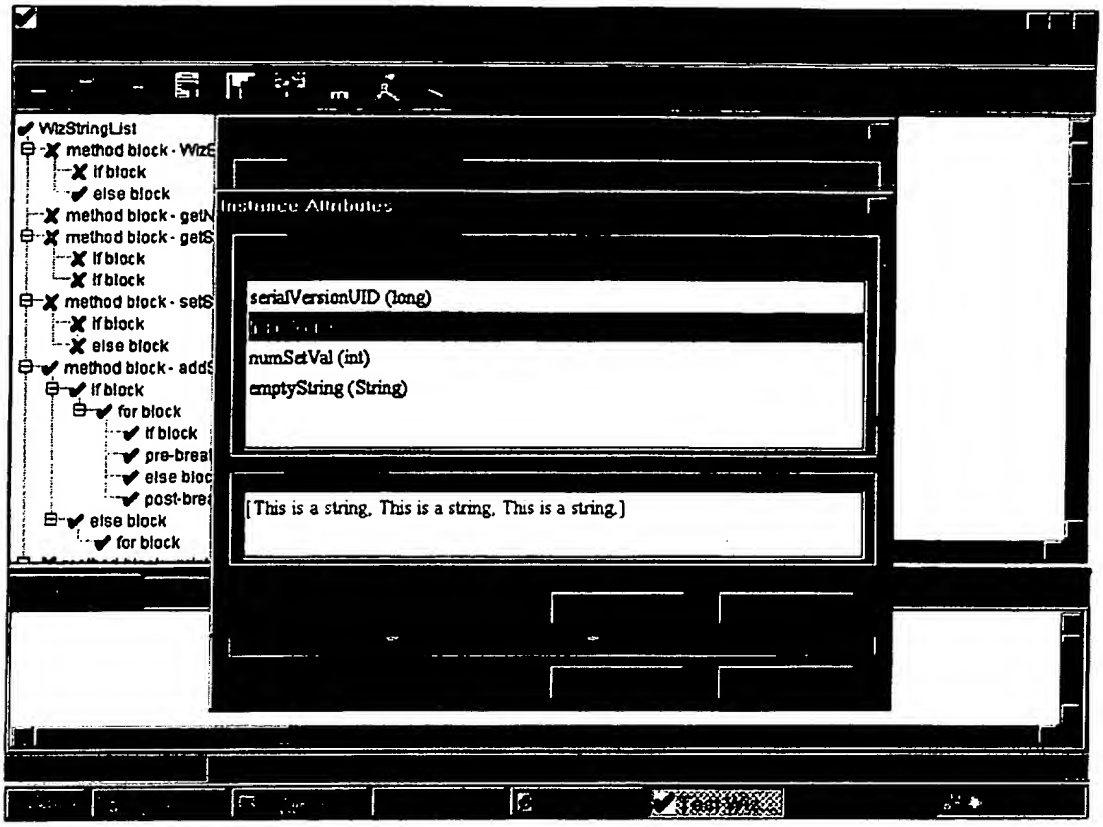


Figure 11A

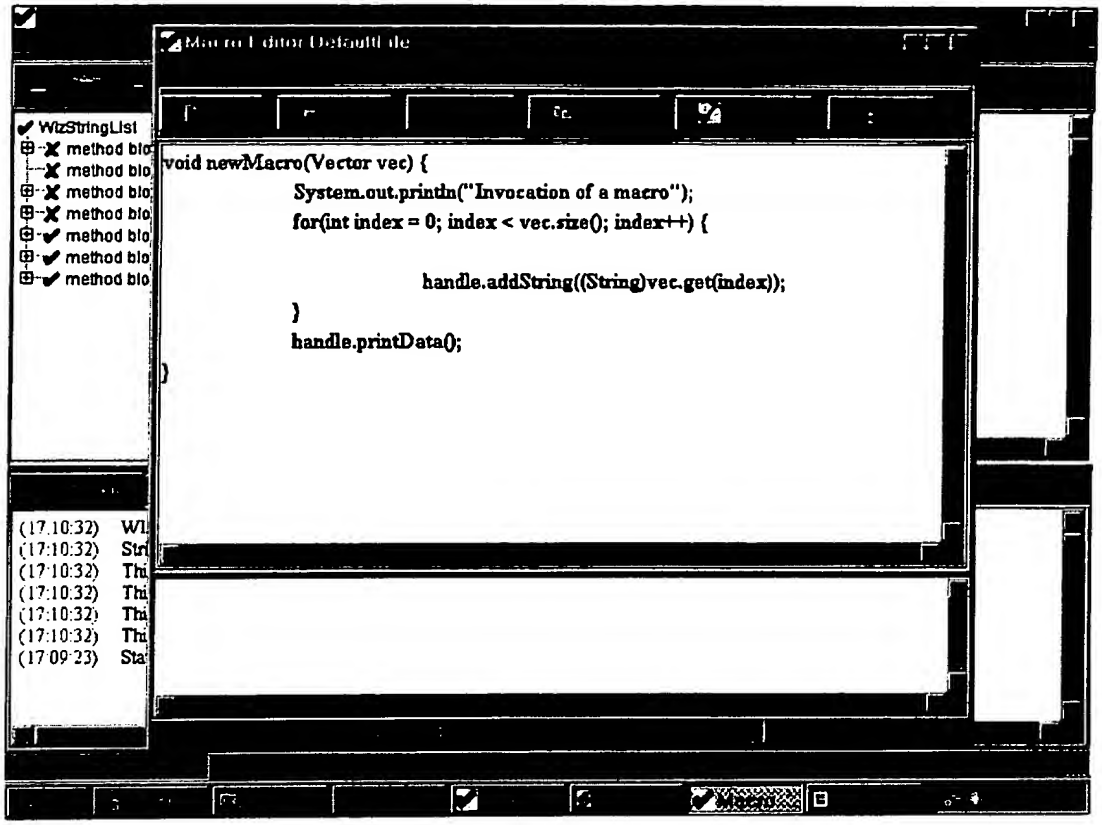

WizStringList
method blo
method blo
method blo
method blo
method blo
method blo
method blo
void newMacro(Vector vec) {
System.out.println("Invocation of a macro");
for(int index = 0; index < vec.size(); index++) {
handle.addString((String)vec.get(index));
}
handle.printData();
}
(17:10:32)
(17:10:32)
(17:10:32)
(17:10:32)
(17:10:32)
(17:10:32)
(17:09:23)

Figure 11B

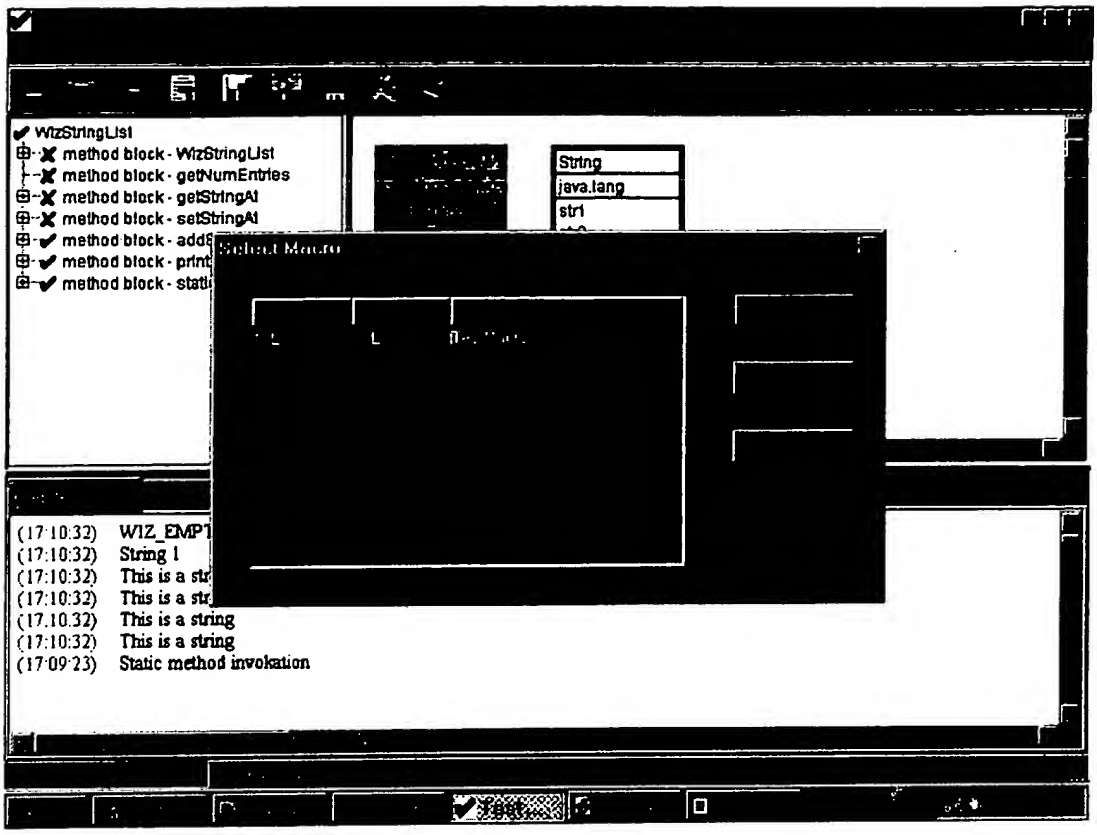

WizStringList
method block - WizStringList
method block - getNumEntries
method block - getStringAt
method block - setStringAt
String
java.lang
(17:10:32) WIZ_EMPT
(17:10:32) String 1
(17:10:32) This is a str
(17:10:32) This is a str
(17:10:32) This is a string
(17:10:32) This is a string
(17:09:23) Static method invokation

Figure 11C